추상 팩토리**(Abstract Factory)**와 팩토리 메소드**(Factory Method)** 패턴 정리

1. 추상 팩토리**(Abstract Factory)** 패턴

1.1 개념

- 추상 팩토리 패턴은 관련된 객체들의 집합을 생성하는 인터페이스를 제공하는 패턴이다.
- 구체적인 클래스의 인스턴스를 만들지 않고, 동일한 주제의 객체들을 생성하는 방법을 제공한다.
- 팩토리 메소드 패턴을 확장한 개념으로, 여러 개의 팩토리를 묶어 일관된 객체 생성을 보장한다.

1.2 구조

- 추상 팩토리**(Abstract Factory)**: 구체적인 팩토리 클래스에서 구현할 인터페이스를 정의한다.
- 구체적 팩토리**(Concrete Factory)**: 인터페이스를 구현하여 특정 제품군을 생성한다.
- 추상 제품**(Abstract Product)**: 생성될 제품의 공통 인터페이스를 정의한다.
- 구체적 제품**(Concrete Product)**: 실제 생성될 객체를 의미하며, 인터페이스를 구현한다.

2. 팩토리 메소드**(Factory Method)** 패턴

2.1 개념

- 팩토리 메소드 패턴은 객체 생성을 서브클래스에서 처리하도록 하는 패턴이다.
- 객체 생성의 책임을 서브클래스로 넘기고, 상위 클래스에서는 인터페이스만 제공한다.

2.2 구조

- 제품**(Product)**: 생성될 객체의 공통 인터페이스를 정의한다.
- 구체적 제품**(Concrete Product)**: 실제 생성될 객체를 구현한다.
- 창조자**(Creator)**: 팩토리 메소드를 정의하는 인터페이스를 제공한다.
- 구체적 창조자**(Concrete Creator)**: 팩토리 메소드를 오버라이드하여 객체를 생성한다.

3. 추상 팩토리 **vs** 팩토리 메소드

비교 항목	추상 팩토리 패턴	팩토리 메소드 패턴
객체 생성 방식	관련 객체군을 생성하는 팩토리를 제공	단일 객체 생성을 위한 메소드 제공
클래스 계층구조	여러 개의 구체적 팩토리를 포함	하나의 팩토리 메소드가 서브클래스에서 구현됨
확장성	새로운 제품군 추가가 어렵지만, 기존 제품 확장은 쉬움	새로운 제품 추가는 가능하지만, 기존 팩토리를 수정해야 할 수도 있음
사용 사례	UI 컴포넌트, 데이터베이스 연결 등 다양한 객체군을 생성할 때	특정 객체 생성 로직을 캡슐화할 때

4. 결론

- 팩토리 메소드 패턴은 객체 생성을 서브클래스에서 결정하도록 하여 확장성을 높인다.
- 추상 팩토리 패턴은 관련된 객체군을 생성하는 데 유용하며, 여러 제품군이 필요할 때 적합하다.
- 프로젝트의 요구 사항에 따라 적절한 패턴을 선택하는 것이 중요하다.